Computational Science - ITB

Ridlo W. Wibowo || 20912009

November 3, 2012

Problem. Buatlah program untuk solusi Parabolic Partial Differential Equations (PDE),
- *Forward Difference Method* (FTCS)
- *Backward Difference Method* (BTCS)
- *Crank-Nicolson Method* (C-N)

Lalu tentukan solusi untuk PDE:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{0.1}$$

untuk $0 < x < \pi$ dan $t > 0$ dengan,

- syarat batas:
  $u(0, t) = u(\pi, t) = 0.0$ untuk $t > 0$

- syarat awal:
  $u(x, 0) = \sin(x)$ untuk $0 \le x \le \pi$

Gunakan $h = \frac{\pi}{10}$ dan $k = 0.05$ dan bandingkan hasilnya untuk $t = 0.5$.
(solusi eksak $u(x, t) = e^{-t}\sin x$)

Result. Dari penurunan dan algoritma yang diberikan di buku *Numerical Analysis* oleh Richard L.Burden dan J. Douglas Faires, lalu dapat diterapkan untuk membuat program penyelesaian masalah PDE. Program yang telah dibuat terlampir di akhir (*ftcs.cpp*, *btcs.cpp*, *CN.cpp*).

- Tabel FTCS

x_i	$u_{(x_i,0.5)}$	$w_{(x_i,0.5)}$	$\|u-w\|_{(x_i,0.5)}$
0	0	0	0
0.314159	0.187428	0.18582	0.001608
0.628319	0.35651	0.35345	0.00306
0.942478	0.490694	0.486482	0.004212
1.25664	0.576845	0.571894	0.004951
1.5708	0.606531	0.601325	0.005206
1.88496	0.576845	0.571894	0.004951
2.19911	0.490694	0.486482	0.004212
2.51327	0.35651	0.35345	0.00306
2.82743	0.187428	0.18582	0.001608
3.14159	0	0	0

- Tabel BTCS

x_i	$u_{(x_i,0.5)}$	$w_{(x_i,0.5)}$	$\|u-w\|_{(x_i,0.5)}$
0	0	0	0
0.314159	0.187428	0.190452	0.003024
0.628319	0.35651	0.362261	0.005751
0.942478	0.490694	0.498609	0.007915
1.25664	0.576845	0.58615	0.009305
1.5708	0.606531	0.616315	0.009784
1.88496	0.576845	0.58615	0.009305
2.19911	0.490694	0.498609	0.007915
2.51327	0.35651	0.362261	0.005751
2.82743	0.187428	0.190452	0.003024
3.14159	0	0	0

- Tabel Crank-Nicolson

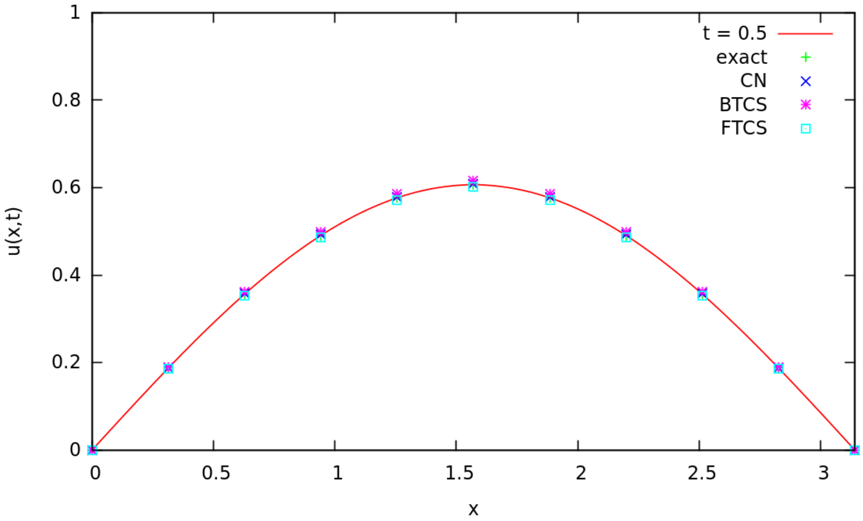
x_i	$u_{(x_i,0.5)}$	$w_{(x_i,0.5)}$	$\|u-w\|_{(x_i,0.5)}$
0	0	0	0
0.314159	0.187428	0.188179	0.000751
0.628319	0.35651	0.357938	0.001428
0.942478	0.490694	0.492659	0.001965
1.25664	0.576845	0.579155	0.00231
1.5708	0.606531	0.60896	0.002429
1.88496	0.576845	0.579155	0.00231
2.19911	0.490694	0.492659	0.001965
2.51327	0.35651	0.357938	0.001428
2.82743	0.187428	0.188179	0.000751
3.14159	0	0	0

Perbandingan Error

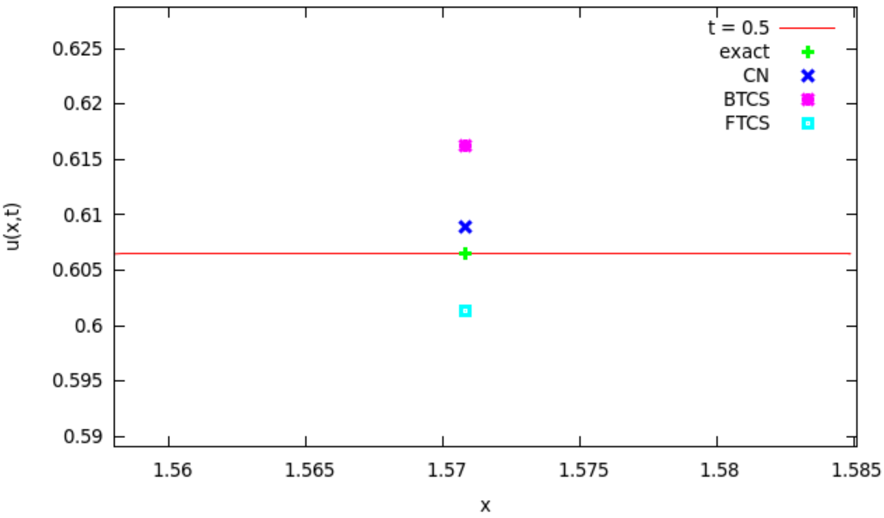
x_i	ftcs	btcs	CN
0.314159	0.001608	0.003024	0.000751
0.628319	0.00306	0.005751	0.001428
0.942478	0.004212	0.007915	0.001965
1.25664	0.004951	0.009305	0.00231
1.5708	0.005206	0.009784	0.002429
1.88496	0.004951	0.009305	0.00231
2.19911	0.004212	0.007915	0.001965
2.51327	0.00306	0.005751	0.001428
2.82743	0.001608	0.003024	0.000751
max:	0.005206	0.009784	0.002429

Dengan nilai k dan h sesuai persoalan untuk $t = 0.5$ ternyata ketiga metode konvergen, dan yang paling baik adalah *Crank-Nicolson*. Metode FTCS dengan nilai $\lambda = 0.506606$ (> 0.5), ternyata masih konvergen.

Kondisi Akhir ($t = 0.5$)



Plot hasil akhir ($t = 0.5$).



Diperbesar disekitar $x = \frac{\pi}{2}$ ($t = 0.5$).

LAMPIRAN

ftcs.cpp

```
/**********************************************/
/* ftcs.cpp (forward time central space)      */
/* forward difference method - parabolic PDE  */
/* Copyleft (c) 2012. Ridlo W. Wibowo         */
/**********************************************/
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#define _USE_MATH_DEFINES
using namespace std;

int main(){
    cout << "### Parabolic PDE - Forward Difference Method ###\n";
    cout << "--- Equation : d(u)/dt = d^2(u)/dx^2 with u(x,t) ---\n";
    double xi = 0.0, xf=M_PI; // L=1.0 -> rentang x
    double ti = 0.0, tf=0.5; // time
    int nt = 10; // jumlah pemenggalan di t
    int nx = 10; // jumlah pemenggalan di x

    double k = (tf-ti)/nt;
    double h = (xf-xi)/nx;

    double alpha = 1.;
    double lam = alpha*alpha*k/(h*h);
    cout << "alpha         = " << alpha << endl;
    cout << "step in x (h) = " << h << endl;
    cout << "step in t (k) = " << k << endl;
    cout << "lambda        = " << lam << endl;

    double wi[100];
    double wf[100];
    double x[100];

    // syarat batas
    wi[0] = 0.0;
    wi[nx] = 0.0;

    // syarat awal
    x[0] = xi;
    x[nx] = xf;
    for (int i=1; i<nx; i++){
        x[i] = x[i-1] + h;
        wi[i] = sin(x[i]);}

    // bentuk output filenya aneh, karena untuk
    // mempermudah ketika membuat animasi memakai gnuplot
    ofstream out("ftcs-out.txt");
    // print kondisi awal
    for (int i=0; i<=nx; i++){
        out << x[i] << " " << wi[i] << "\n";}

    // kerjo dimulai.. ayo
    for (int j=1; j<=nt; j++){
        wf[0] = wi[0];
        wf[nx] = wi[nx];
        for (int i=1; i<nx; i++){
            wf[i] = (1.-2.*lam)*wi[i] + lam*(wi[i+1] + wi[i-1]);}

        // print, njur sekalian tuker baru
        out << "\n\n" ;
        for (int i=0; i<=nx; i++){
            out << x[i] << " " << wf[i] << "\n";
            wi[i] = wf[i];}}
    out.close();
    cout << "finish...\n";
    return 0;
}
```

btcs.cpp

```
/*********************************************/
/* btcs.cpp (backward time central space)     */
/* backward difference method - parabolic PDE */
/* Copyleft (c) 2012. Ridlo W. Wibowo         */
/*********************************************/
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#define _USE_MATH_DEFINES
using namespace std;


int main(){
    cout << "### Parabolic PDE - Backward Difference Method ###\n";
    cout << "--- Equation : d(u)/dt = d^2(u)/dx^2 with u(x,t) ---\n";
    double xi = 0.0, xf=M_PI; // L=1.0 -> rentang x
    double ti = 0.0, tf=0.5; // time
    int nt = 10; // jumlah pemenggalan di t
    int nx = 10; // jumlah pemenggalan di x

    double k = (tf-ti)/nt;
    double h = (xf-xi)/nx;

    double alpha = 1.;
    double lam = alpha*alpha*k/(h*h);
    cout << "alpha        = " << alpha << endl;
    cout << "step in x (h) = " << h << endl;
    cout << "step in t (k) = " << k << endl;
    cout << "lambda       = " << lam << endl;

    double w[100];
    double x[100];
    double l[100], u[100], z[100];

    // syarat batas
    w[0] = 0.0;
    w[nx] = 0.0;

    // syarat awal, insiasi
    x[0] = xi;
    x[nx] = xf;
    for (int i=1; i<nx; i++){
        x[i] = x[i-1] + h;
        w[i] = sin(x[i]);}

    // Crout method
    l[1] = 1. + 2.*lam;
    u[1] = -lam/l[1];

    for (int i=2;i<(nx-1);i++){
        l[i] = 1. + 2.*lam + lam*u[i-1];
        u[i] = -lam/l[i];}

    l[nx-1] = 1. + 2.*lam + lam*u[nx-2];

    //for (int i=1;i<nx;i++){
    //     cout << l[i] << endl;}

    // bentuk output filenya aneh, karena untuk
    // mempermudah ketika membuat animasi memakai gnuplot
    ofstream out("btcs-out.txt");
    // print kondisi awal
    for (int i=0; i<=nx; i++){
        out << x[i] << " " << w[i] << "\n";}

    // kerjo dimulai.. ayo
    for (int j=1; j<=nt; j++){
        w[0] = 0.0;
        w[nx] = 0.0;
        z[1] = w[1]/l[1];
        for (int i=2;i<nx;i++){
            z[i] = (w[i] + lam*z[i-1])/l[i];}
        w[nx-1]=z[nx-1];

        for (int i=nx-2;i>=1;i--){
            w[i] = z[i] - u[i]*w[i+1];}

        out << "\n\n";
        for (int i=0;i<=nx;i++){
            out << x[i] << " " << w[i] << "\n";}}
    out.close();

    cout << "finish...\n";
    return 0;
}
```

CN.cpp

```
/************************************************/
/* CN.cpp (Crank-Nicolson method)               */
/* parabolic PDE                                */
/* Copyleft (c) 2012. Ridlo W. Wibowo           */
/************************************************/
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#define _USE_MATH_DEFINES
using namespace std;

int main(){
    cout << "### Parabolic PDE - Crank-Nicolson Method ###\n";
    cout << "--- Equation : d(u)/dt = d^2(u)/dx^2 with u(x,t) ---\n";
    double xi = 0.0, xf=M_PI; // L=1.0 -> rentang x
    double ti = 0.0, tf=0.5; // time
    int nt = 10; // jumlah pemenggalan di t
    int nx = 10; // jumlah pemenggalan di x

    double k = (tf-ti)/nt;
    double h = (xf-xi)/nx;

    double alpha = 1.;
    double lam = alpha*alpha*k/(h*h);
    cout << "alpha         = " << alpha << endl;
    cout << "step in x (h) = " << h << endl;
    cout << "step in t (k) = " << k << endl;
    cout << "lambda        = " << lam << endl;

    double w[100];
    double x[100];
    double l[100], u[100], z[100];

    // syarat batas
    w[0] = 0.0;
    w[nx] = 0.0;

    // syarat awal, insiasi
    x[0] = xi;
    x[nx] = xf;
    for (int i=1; i<nx; i++){
        x[i] = x[i-1] + h;
        w[i] = sin(x[i]);}

    // Crout method
    l[1] = 1. + lam;
    u[1] = -lam/(2.*l[1]);

    for (int i=2;i<(nx-1);i++){
        l[i] = 1. + lam + lam*u[i-1]/2.;
        u[i] = -lam/(2.*l[i]);}

    l[nx-1] = 1. + lam + lam*u[nx-2]/2.;

    //for (int i=1;i<nx;i++){
    //     cout << l[i] << endl;}

    // bentuk output filenya aneh, karena untuk
    // mempermudah ketika membuat animasi memakai gnuplot
    ofstream out("CN-out.txt");
    // print kondisi awal
    for (int i=0; i<=nx; i++){
        out << x[i] << " " << w[i] << "\n";}

    // kerjo dimulai.. ayo
    for (int j=1; j<=nt; j++){
        w[0] = 0.0;
        w[nx] = 0.0;
        z[1] = ((1.-lam)*w[1] + (lam/2.)*w[2])/l[1];
        for (int i=2;i<nx;i++){
            z[i] = ((1.-lam)*w[i] + (lam/2.)*(w[i+1] + w[i-1] + z[i-1]))/l[i];}
        w[nx-1]=z[nx-1];

        for (int i=nx-2;i>=1;i--){
            w[i] = z[i] - u[i]*w[i+1];}

        out << "\n\n";
        for (int i=0;i<=nx;i++){
            out << x[i] << " " << w[i] << "\n";}}
    out.close();
    cout << "finish...\n";
    return 0;}
```